given, and actions are represented by a program that generates complete state descriptions. Therefore, all state representations are complete. In most problems, a state is a simple data structure: a permutation of the pieces in the eight puzzle, the position of the agent in a route-finding problem, or the position of the six people and the boat in the missionaries and cannibals problem. State representations are used only for successor generation, heuristic function evaluation, and goal testing.

- **Representation of goals.** The only information that a problem-solving agent has about its goal is in the form of the goal test and the heuristic function. Both of these can be applied to states to decide on their desirability, but they are used as "black boxes." That is, the problem-solving agent cannot "look inside" to select actions that might be useful in achieving the goal.
- **Representation of plans.** In problem solving, a solution is a sequence of actions, such as "Go from Arad to Sibiu to Fagaras to Bucharest." During the construction of solutions, search algorithms consider only unbroken sequences of actions beginning from the initial state (or, in the case of bidirectional search, ending at a goal state).

Let us see how these design decisions affect an agent's ability to solve the following simple problem: "Get a quart of milk and a bunch of bananas and a variable-speed cordless drill." Treating this as a problem-solving exercise, we need to specify the initial state: the agent is at home but without any of the desired objects, and the operator set: all the things that the agent can do. We can optionally supply a heuristic function: perhaps the number of things that have not yet been acquired.

Figure 11.2 shows a very small part of the first two levels of the search space for this problem, and an indication of the path toward the goal. The actual branching factor would be in the thousands or millions, depending on how actions are specified, and the length of the solution could be dozens of steps. Obviously, there are too many actions and too many states to consider. The real difficulty is that the heuristic evaluation function can only choose among states to decide which is closer to the goal; it cannot eliminate actions from consideration. Even if the evaluation function could get the agent into the supermarket, the agent would then resort to a guessing game. The agent makes guesses by considering actions—buying an orange, buying tuna fish, buying corn flakes, buying milk—and the evaluation function ranks these guesses—bad, bad, bad, good. The agent then knows that buying milk is a good thing, but has no idea what to try next and must start the guessing process all over again.

The fact that the problem-solving agent considers sequences of actions starting from the initial state also contributes to its difficulties. It forces the agent to decide first what to do in the initial state, where the relevant choices are essentially to go to any of a number of other places. Until the agent has figured out *how* to obtain the various items—by buying, borrowing, leasing, growing, manufacturing, stealing—it cannot really decide where to go. The agent therefore needs a more flexible way of structuring its deliberations, so that it can work on whichever part of the problem is most likely to be solvable given the current information.

The first key idea behind planning is to *"open up" the representation of states, goals, and actions.* Planning algorithms use descriptions in some formal language, usually first-order logic or a subset thereof. States and goals are represented by sets of sentences, and actions are represented by logical descriptions of preconditions and effects. This enables the planner to make